AD-A214 418

Integrated Interfaces

Final Technical Report*
DARPA Order No. 5941
Contract N00014-87-K-0130
1/1/87 - 6/30/89

DTIC
S ELECTE D
NOV 1 3 1989
D

Yigal Arens
University of Southern California
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
(213) 822-1511 ext. 766

Abstract

This report summarizes the major results of work done over the term of the Integrated Interfaces (II) project at USC/ISI.

The construction of the user interface is extremely costly, with some estimates as high as 60% of the development of a software system. Compounding the problem is the large number of interface modalities which exist (graphics, forms, natural language, etc.) and which the end-user would like to have available. ISI chose an actual Navy command and control briefing task which is currently performed manually at considerable cost. The Integrated Interface (II) project developed models of the domain of the naval application, and of the capabilites of the various interface modalities available. With a relatively small collection of rules, a multi-modal interface to the task was generated. It responds to user-requests dynamically, taking into account the particular data to be displayed and its representation of the semantics of the domain and the user's request. The generation of briefing maps which takes several hours manually can be done by II in minutes. II has demonstrated that advanced multi-modal interfaces in appropriate domains can be generated automatically using AI techniques. (KR)

*The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

89 11 08 050

1 Problem: Produce Multi-Modal, Intelligent Interfaces

The cost of developing human-computer interfaces is very high. It is common to find over 50 percent of the code of interactive systems devoted to interface tasks [Sutton78]. It is also clear that. in order to control costs. interfaces are "simplified" and the applications made harder to use than is necessary.

Approximately five years ago, the phrase **User Interface Management System (UIMS)** was coined to describe the most ambitious efforts to ease user interface development. A UIMS has been described as a tool that

- provides a software developer support for the definition of the user/application dialogue
- imposes external control on the application
- provides support for the presentation of the application's output
- includes an interactive component providing support for the interaction between an application and an end user [Betts87]

Unfortunately, current UIMSs are too limited to support complex tasks that require integration of several interface modes, such as natural language, graphics, tables, forms, and maps. This is the case even when concentrating on output presentations, as we did with the Integrated Interfaces (II) project. Complex, integration requiring tasks existed in the domain in which we chose to demonstrate our work — the preparation of daily briefing maps describing the location, course, employment, and more for all ships in the Pacific Fleet. For example, inspection of a briefing map as it is currently prepared manually for the Admiral of the Pacific Fleet (Figure 1) reveals the integration of a map of the relevant region. graphical icons representing ships, natural language text describing ship employments, and a table listing ships in port.

By itself, the availability of several interface modes does not suffice for us to consider an interface truly intelligent and multi-modal. The interface modalities must be *integrated.* By this we mean that different items of information must be distributed to appropriate modes, the amount of redundancy should be limited, and different presentation modes must work from the same meaning representation to assure coordination and agreement among portions of the display. For example, the Navy map requires different modes for different information. e. g., the position of ships at sea is presented through placement on the map, current heading of ships that are en route is represented by orientation of an icon, sailing plan is represented by natural language text, etc. None of this information is duplicated. Co-reference is established by lines connecting text, labels, and icons. Furthermore, since pre-designed multi-modal displays are not sufficient for a constantly changing environment, the interface system integrating a set of modes must produce displays *dynamically,* taking into account the specific amount and

QUALITY INSPECTED 2

Codes
Dist | Avail and/or Special
A-1

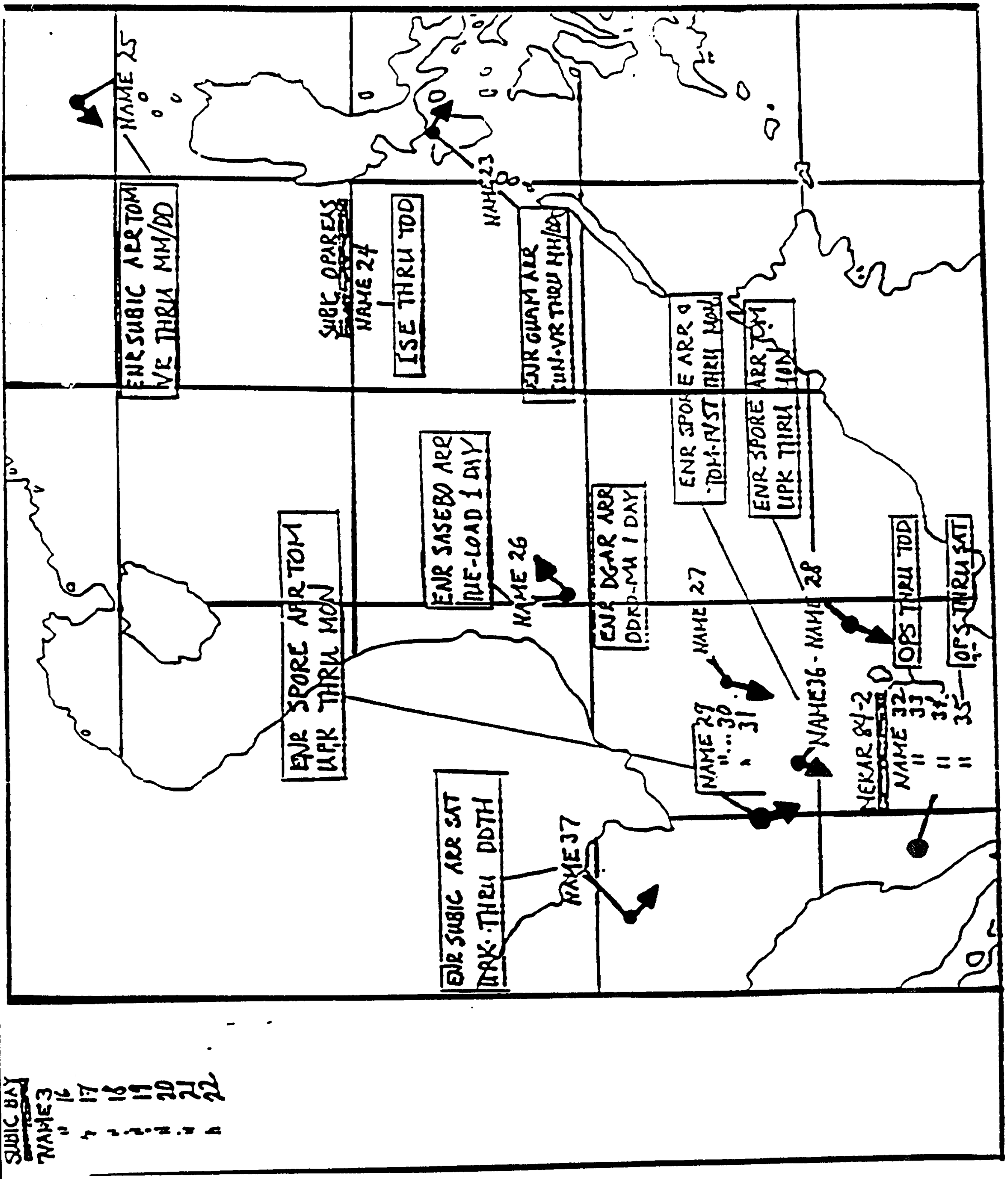


Figure 1: Example of Manually Produced CINCPACFLT Briefing Report (unclassified).

nature of the data and the context in which it is being shown to the user. Finally, the interface must support *generalization* and *enhancement*. since it is certain to require modification overtime.

Existing multi-modal systems do not and cannot achieve these objectives. Typically, they are hand-coded, with the choice of modalities pre-determined by the interface designer.

2 The Integrated Interfaces' Approach

The problems described above were addressed by the Integrated Interfaces (II) project at ISI. II produced a design that supports integration of display modes. dynamically produces multi-modal displays. and supports generalization and enhancement. It does all of this through a system of models and rules.[1]

The construction of interface displays is enabled in II by the existence of a *model* containing descriptions of the application domain and the capabilities of the various interface modalities. Interface displays are constructed with the application of antecedent-consequent rules associated with the representations of domain information. The rules explicitly state how information described in application terms relates to presentation modes. These rules take advantage of the model of interface capabilities to integrate the modes. Given information to display, an interpreter applies these rules to dynamically produce displays.

Specifically, the display design process involves *realizing* the categories to which a new piece of information belongs, *selecting* the rules appropriate for these categories, and *redescribing* the information in appropriate visual or textual forms.

The Navy's report-generating activity can be described as following a process and rules similar to those encoded in our system. Information concerning ships is realized as belonging to certain known categories — e. g., the ships' planned activities. Rules for translating such information into components of a report — e. g., drawing an icon on a map, or using text — are then examined, and a rule most appropriate is selected. The information about the ships is then redescribed as part of the presentation being prepared.

[1]More information about the II project, including examples of its operation may be found in [Arens88].

3 Technology Used

3.1 Model-Driven Programming

Our method for developing and operating the user interface is referred to as **model-driven programming**[Yen88]. The essence of the approach has a cyclical control regimen analogous to, but more powerful than that used in rule-based systems. A model-driven programming system maintains a hierarchical lattice of abstractions representing a **model** of the conceptual objects in a task domain. The model includes:

- Specifications of objects and their classes,
- Relationships among them.
- Behaviors they may engage in. and
- The effects of such behaviors.

This provides a natural way to conceptualize problems, allowing people who are inexperienced with the system. but who understand the domain, to provide help with the programming.

Actions are associated with classes of objects in the model. A processing cycle is initiated by the introduction of new data generated by events within the system or from its interface to the outside world. The system generates descriptions of the new data, determines which class or classes in the model the data is a member of, makes additional data assertions implied by the classification, and triggers behavior specified as associated with the particular classes. Thus, the approach blends elements of rule-based systems, object-oriented systems, and truth maintenance facilities.

For example, JFK is an individual ship, a member of the class *Aircraft Carrier*. That class is a subclass of *Surface Ship*, which is a subclass of *Ship*, etc. A *Sighting* is a relationship between ships. With each of these classes one may associate actions in the model, among them the actions describing how to display them on a map.

The following subsections in this section describe our models and the actions (the rules of II) in more detail. The last subsection provides information about the specific knowledge representation tools we use in the Integrated Interfaces project.

3.2 Modeling

Our model characterizes the categories of *entities* with which our user interface deals. There is actually a single model containing all entities and describing the relations among them. For convenience, however, we have chosen to use the terms **Application Model** and the **Interface Model**. The former term refers to the collection of entities specific to the domain of the application to which we are interfacing. The latter refers to the collection of entities of general concern when specifying an interface. Despite our terminology, one should keep in mind that the dividing line between the two collections is not a sharp one, and that there exist relationships involving entities in both collections, as we shall see below.

When converting the interface to a new application, the existing interface model can be reused, and need not be rewritten. Although some sharing may be possible with the application model as well, considerable work on a model for the new application will probably be necessary. The interface designer will be aided, however, by the higher level structure of the model, containing such abstract concepts as *action*, *event*, etc. These existing concepts can be used to help guide and organize the design of a model for a new application.

3.2.1 Application Model

This portion of the model identifies the categories of objects and actions in a common-sense view of the application domain of our system. We indicate subclass relations present between categories, as well as relationships between between objects and actions. For the Navy application we have been working with, we include the various categories of ships and sailing activities, as well as geographical information about the Pacific region. We also include specific knowledge, such as that a *Disabled Ship* is a type of *Ship*, and that a *Repair* activity involves a *Disabled Ship*.

For a graphical representation of a small portion of the application model see Figure 2.

3.2.2 Interface Model

Another portion of the model describes the categories of objects and actions in the interface world. The objects here include tables and forms and their structure (columns, rows, headings), maps, text strings, icons, windows, colors, and so forth. The actions include the creation and deletion of displays and natural language text, and various user requests.

The interface model provides a uniform view of all modes to the interface. As a consequence of the standardized representation of interface modes, an inspection of the model can tell the system what the capabilities of any particular interface mode are. Data to be presented is described using the model's terminology in a unique way, regardless of the modality that will be used to display it to the user. It is this feature of our approach that is largely responsible for the ability to delay decisions about the modalities to be used until run-time — i. e., for II's dynamic response.

For a graphical representation of a small portion of the interface model see Figure 3.

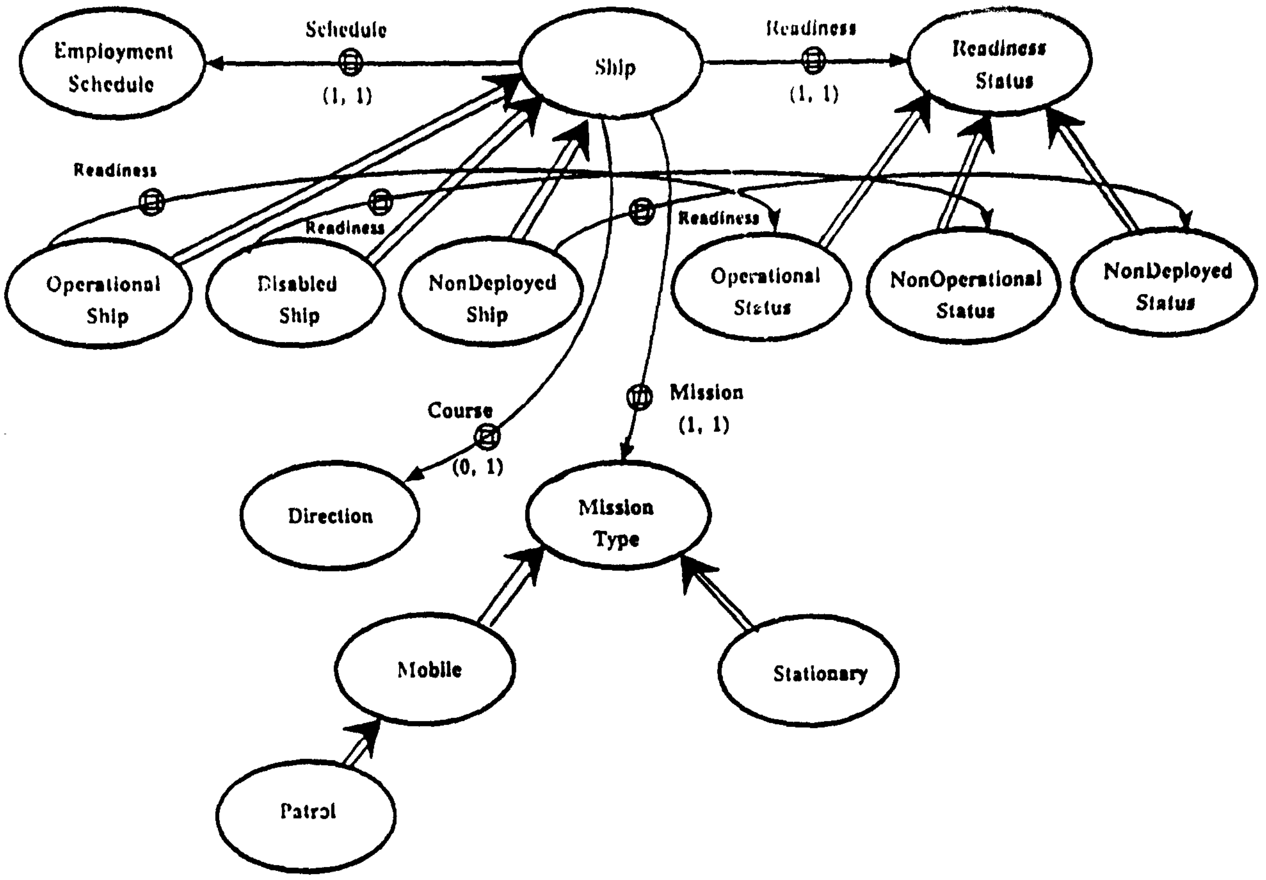


Figure 2: Fragment of Domain Model Containing *Ship*.

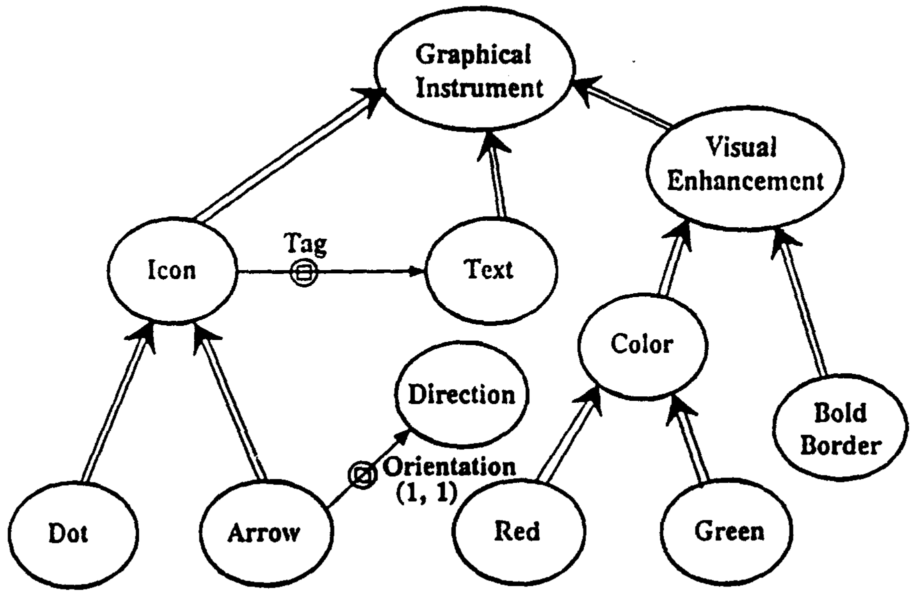


Figure 3: Fragment of Interface Model Containing *Graphical-Instrument*.

3.3 Rules

Generally speaking the presentation rules map objects from the application domain model into objects in the interface model. An entity that describes a daily status report may be mapped into a map. A position report may be mapped onto an icon on the map. A ship's planned activities may be mapped into a natural language string.

The following is a paraphrase of part of a rule for the CINCPACFLT briefing application: "To display an *Enroute.Ship*, use an *Arrow* as its *Shape*, with its *Course* establishing the arrow's *Orientation*, and *Text* as a *Tag* presenting its *Sailing Plan.*" As can be seen this rule takes its condition from the application model and the rest of its vocabulary from the application and interface models.

It is the rules in conjunction with the interface model, that allow integration. They can be used to distribute information among modes. minimize redundancy, and coordinate presentations. For example, the above rule creates an mixed graphic and natural language text display.

3.4 Presentation Agents

II has three output systems:

- The Geographic Display Agent (GDA), a map graphics system developed at ISI,
- Penman, a text generator [Sondheimer86], and
- The QForms forms kit [Kaczmarek84].

As stated above. the representations produced by the application of rules are not actually the commands to these output systems. Instead, they are interpreted by subsystems we call **Presentation Agents.** The Agents can be thought of as device drivers. They translate structures using interface model terms into calls to the desired device.

One assures that any requested output structure can actually be created by the desired device, by ascertaining that the interface model contains only such creatable structures. For example, that part of the interface model hierarchy which contains information about forms. includes only representations of forms that QForms can generate.

The incorporation of a new mode into II thus involves the addition of modeling information describing displays produced by the new mode, as well as the design of a presentation agent for it.

3.5 Knowledge Representation Tools

Our implementation of presentation design by II depends on two knowledge representation systems: NIKL and KL-TWO. NIKL holds our models. KL-TWO automatically carries out realization. KL-TWO also holds the demands for presentation and receives the forms read by the device drivers. This section provides a brief introduction to these tools.

3.6 Model Implementation: NIKL

NIKL [Kaczmarek86] is a network knowledge-base system descended from KL-ONE [Brachman85]. This type of system supports description of the entities that make up a domain. The central components of the notation are sets of concepts and roles, organized in IS-A hierarchies. These hierarchies identify when membership in one category (or the holding of one relationship) entails membership in (or the holding of) another. The roles are associated with concepts (as *role restrictions*), and identify the relationships that can hold between individuals that belong to the categories. The role restrictions also hold number restrictions on the entities that fill these roles.

We have been experimenting with a naval assets model for the naval briefing application mentioned above. The model has a concept *Disabled-Ship* that is meant to identify ships that are unable to carry out their missions. A *Disabled-Ship* IS-A type of *Ship* distinguished from *Ship* by having a role restriction *Readiness* that relates *Disabled-Ship* to *NonOperational-Status*, i.e., all ships with nonoperational status are disabled. All *Ships* can have exactly one filler of the *Readiness* role restriction. The concept of *NonOperational-Status* is partly defined through the IS-A relation to a concept *Readiness-Status*. This situation is shown graphically in Figure 2 in a notation used for KL-ONE knowledge bases.

In flavor, NIKL is a frame system, with the concepts equivalent to frames and the role restrictions to slots. However, the NIKL representation has a formal semantics. We could translate our NIKL knowledge bases into predicate calculus expressions and use a theorem prover to make the same inferences we do. However, NIKL is optimized for the limited inferences it makes, and a general purpose theorem prover would be less efficient.

3.7 Rule Implementation: KL-TWO

KL-TWO is a hybrid knowledge representation system that takes advantage of NIKL's formal semantics [Vilain85]. KL-TWO links a reasoner, PENNI, to NIKL. PENNI, an enhanced version of RUP [McAllester82], reasons using propositional logic. It is more restricted than systems that use first order logic and a general purpose theorem prover.

PENNI manages a data base of propositions of the form (P a) and (Q a b), where the forms are variable free. The first item in each ordered pair is the name of a concept in an associated NIKL network, and the first item in each ordered triple is the name of a role in the network. The assertion of any form (P a) is a statement that the individual a is a thing described by the concept P. The assertion (Q a b) states that individuals a and b are related by the abstract relation described by Q.

NIKL adds to PENNI the ability to do taxonomic reasoning. Assume the NIKL database contains the concepts described above. Assume that we assert just the following three facts: (*Ship Sprite*), (*Readiness Sprite C4*) and (*NonOperational-Status C4*) (*C4* is a U.S. Navy readiness code). Using the knowledge base, PENNI is able to deduce that any *Ship* whose *Readiness* is a *NonOperational-Status* is a *Disabled-Ship*. So if we ask if (*Disabled-Ship Sprite*) is true. KL-TWO will reply positively.

PENNI also provides a truth maintenance system that keeps track of the facts used to deduce others. When our rules are used to determine aspects of a presentation from facts about the world. the truth maintenance system records the dependencies between the domain and the presentation. For example. (*Readiness Sprite C4*) triggers a rule that asserts (*Disabled-Ship Sprite*). If (*Readiness Sprite C4*) is retracted. PENNI's truth maintenance system will automatically retract the assertion that the Sprite is a disabled ship.

4 Status of Integrated Interfaces Project

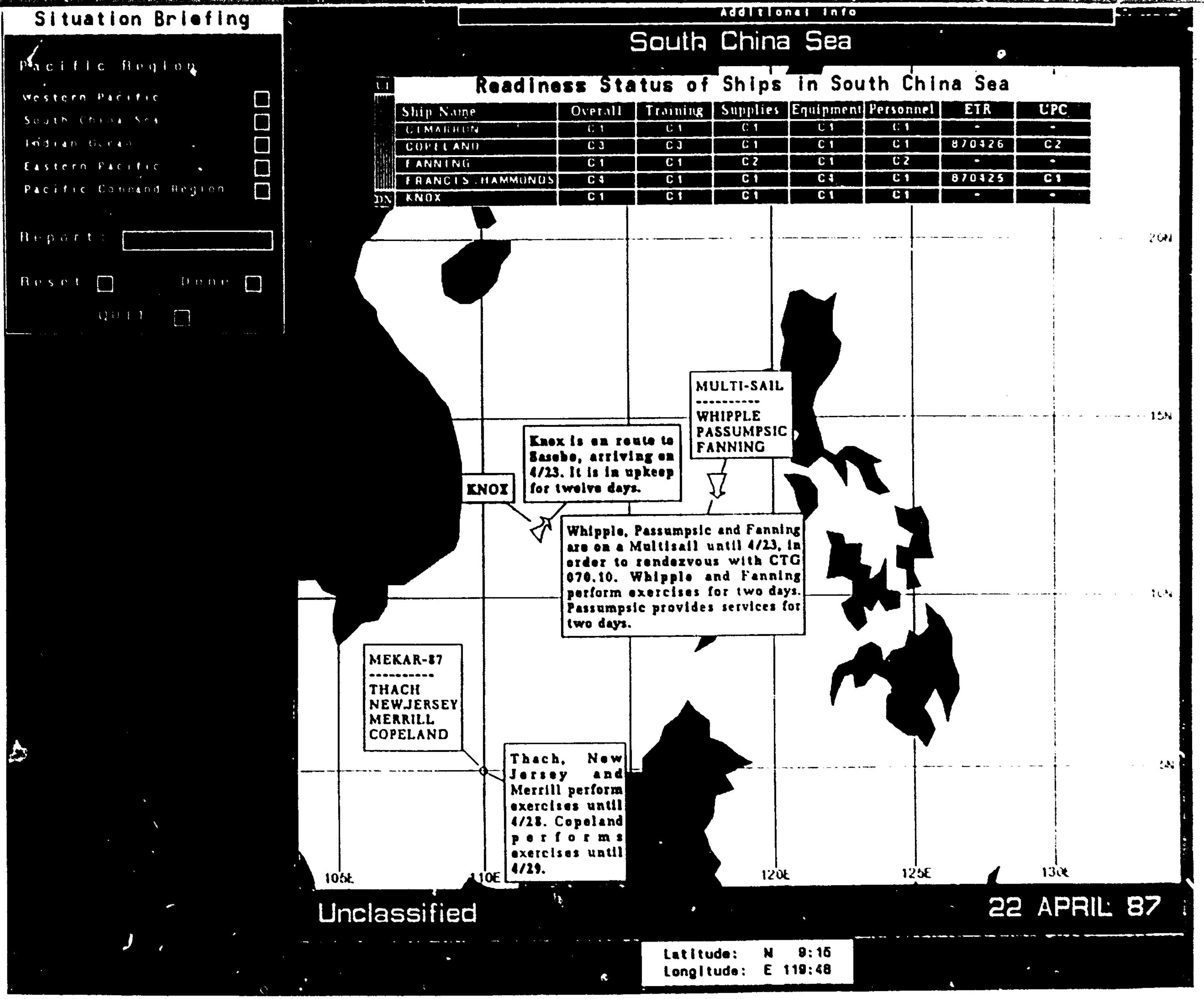
DARPA responded to the limitations of existing systems by seeking a research initiative in user interfaces through a CBD announcement in August 1985. ISI submitted a proposal for this effort and won a two-year contract for the **Integrated Interfaces (II)** project at the beginning of 1987.[2]

During the following two years, ISI designed and implemented an initial version of the Integrated Interfaces system that integrates a variety of modes and demonstrated its application to the task of naval command briefings (See Figure 4 for a black and white example of II system output. Actual out put is in color).

In the existing demonstration system a user may access information about the location, tasks, readiness status, course, and more, of Navy ships in the Pacific Ocean. The retrieved information is displayed using a combination of maps, menus, tables, icons, strings, and natural language output.

The system was written in Common Lisp and runs in the X windows environment under UNIX on HP 9000 Model 350 workstations. Displays are presented on a Renaissance color graphics monitor. The map graphic modality is supported by ISI's Graphic Display Agent. Menus and forms are created using The QForms forms

[2]A short time later. a smaller research effort was established at CUBRC in Buffalo, New York



Readiness Status of Ships in South China Sea

Ship Name	Overall	Training	Supplies	Equipment	Personnel	ETR	UPC
CIMARRON	C1	C1	C1	C1	C1	-	-
COPELAND	C3	C3	C1	C1	C1	870426	C2
FANNING	C1	C1	C2	C1	C2	-	-
FRANCIS HAMMONDS	C4	C1	C1	C4	C1	870425	C1
KNOX	C1	C1	C1	C1	C1	-	-

Knox is en route to Sasebo, arriving on 4/23. It is in upkeep for twelve days.

Whipple, Passumpsic and Fanning are on a Multisail until 4/23, in order to rendezvous with CTG 070.10. Whipple and Fanning perform exercises for two days. Passumpsic provides services for two days.

Thach, New Jersey and Merrill perform exercises until 4/28. Copeland performs exercises until 4/29.

Figure 4: South China Sea Situation Display.

kit [Kaczmarek84]. Natural language output is produced by ISI's Penman system [Sondheimer86].

Detailed plans exist for future work when funds become available.

5 Effectiveness of Implementation and Techniques

We began the Integrated Interfaces research effort with two goals in mind: discovering a method to support the integration of multiple modes and discovering whether a general tool for multi-modal interface construction could be built. We stressed the first goal in the initial phase of the program. In order to support integration, we explored a model-driven programming approach. One portion of the model brought together all interfaces modes. Another portion represented the application for which the interface was being built. Explicit rules were investigated as a way of connected the two models such that a runtime interpreter could apply these rules to produce interface displays. Our research to date has verified that this is a viable approach.

We have realized this design in a system that utilizes several modalities. Specifically, the Integrated Interfaces system can create maps containing icons with string tags and natural language descriptions attached to them. It can further combine such maps with forms and tables presenting additional, related information. In addition, the system is capable of dynamically creating menus for choosing among alternative actions, and more complicated forms for specifying desired information.

We have constructed application models describing concepts in an important real-world domain — the Naval briefing situation. We have implemented rules that enable the creation of different types of integrated multi-modal output displays based on the Navy's current manual practices. We have represented large enough portions of both the general and application specific domains to demonstrate that a model-driven UIMS approach is potentially useful in real-world situations.

Our current effort has resulted in an interface that has generated considerable enthusiasm among CINCPACFLT staff. It has proven to our satisfaction that our model-based approach has considerable utility.

In achieving this result, we have done more than produce a system for constructing and controlling multi-modal application interfaces. We have shown that what would otherwise appear to be distinct communication mechanisms, viz., graphics, natural language, tables, etc., can be treated as part of an integrated whole, all relating to a common level of meaning representation. We have further shown that the decisions on the use of the appropriate mode can be represented straightforwardly by explicit rules relating information presentation situation to the method of presentation. Together, this can serve as the basis of a comprehensive theory of multi-modal communication.

6 Related Work

The literature contains numerous examples of User Interface Management Systems. However, we see our contribution as being our emphasis on *Presentation Planning*, and very few systems are concerned with this aspect of the interface. Perhaps the best known previous work dealing with this issue is that of Mackinlay [1].

Much like part of our system, Mackinlay's *APT* uses information about characteristics of data provided to it, to produce a graphical representation of that data. The differences between the two systems become clear when we consider the variety of data each deals with and the variety of presentations they produce. *APT* produces graphs of various kinds, and much of its effort goes into deciding which axes to choose, and how to indicate the values along each axis. Data dealt with is limited to what can be presented using such graphs. Consequently, Mackinlay has succeeded in producing a system which can generate graphical presentations automatically using only "low-level" information about the objects and their attributes.

Our system is expected to generate a much wider variety of displays, many that would require considerable design work even from an expert human graphic artist.[3] In addition, certain display layouts are often chosen simply to conform to pre-existing preferences of Navy personnel. Consequently, unlike Mackinlay, we are required to provide for the possibility of following pre-set stereotypical instructions in certain cases. We thus must devote considerable effort to recognizing which cases require these special displays.

A further significant difference between the systems is the complexity of the data we are required to present. In order to handle this range of data we must represent it using a sophisticated knowledge representation language, NIKL, a facility which Mackinlay finds unnecessary in *APT*. Both systems make use of sophisticated reasoning facilities.

7 Personnel

The following personnel were supported in full or in part in the duration of this contract:

- Dr. Yigal Arens
- Mr. Chin Y. Chee
- Mr. James Geller
- Dr. Larry Miller
- Mr. Paul Raveling
- Dr. Stuart Shapiro (on sabbatical leave from SUNY Buffalo)
- Dr. Norman K. Sondheimer

[3]As in fact they do. Maps of the kind produced by our system take Navy personnel approximately 4 hours to produce every day.

8 List of Publications

The following publications were written about the work sponsored by under this contract:

1. Arens. Yigal. A Knowledge-Based Multi-Modal Interafce. Submitted to *CAIA-90: IEEE Conference on AI Applications.*

2. Arens. Yigal. A Knowledge-Based Multi-Modal Interafce. Submitted to *AISIG-90: The Annual AI Systems in Government Conference.*

3. Arens. Yigal, C. Chee and P. Raveling. Development of Automatic User Interface Technology: Architecture and Software Engineering Issues. *The Third Annual X Technical Conference.* MIT. Cambridge, Mass. January, 1989.

4. Arens. Yigal, L. Miller, and N. K. Sondheimer. Presentation Planning Using an Integrated Knowledge Base. *Proceedings of the 1988 Workshop on Architectures for Intelligent Interfaces: Elements and Prototypes.* Monterey, California. March 1988.
An expanded version of this paper will appear in a book to be published by Addison-Wessley, title yet to be decided.

5. Arens. Yigal, L. Miller, S. C. Shapiro. and N. K. Sondheimer. Automatic Construction of User-Interface Displays. *AAAI-88: The Seventh National Conference on Artificial Intelligence.* St. Paul, Minnesota. August, 1988. Also available as ISI Technical Report No. ISI/RS-88-218.

6. Miller. Larry, Y. Arens and N. K. Sondheimer. Presentation Planning Using an Integrated Knowledge Base. In *USICON '88: Proceedings of the Third Annual User-System Interface Conference.* Austin, Texas. February, 1988.

7. Miller, Larry, Y. Arens and N. K. Sondheimer. Presentation Planning Using an Integrated Knowledge Base. *SOAR 88: The Second Annual Workshop on Space Operations Automation and Robotics*, NASA Conference Publication 3019. Wright State University, Dayton, Ohio, July 20–23, 1988, pp. 205–218.

References

[Arens88] Arens, Yigal, Lawrence H. Miller, Stuart C. Shapiro and Norman K. Sondheimer. "Automatic Construction of User Interface Displays," Accepted for publication: *Proceedings, AAAI '88*. St. Paul, MN., August, 1988.

[Betts87] Betts, William. "Goals and Objectives for User Interface Software," *Computer Graphics*, 21(2), pp. 73-78, April, 1987.

[Brachman85] Brachman, Ronald J. and James G. Schmolze, "An Overview of the KL-ONE Knowledge Representation System." *Cognitive Science*, 9(2), pp. 171-216, 1985.

[Kaczmarek84] Kaczmarek, T., "CUE Forms Description." ISI Internal Report, USC/ISI, Marina del Rey, CA., 1984.

[Kaczmarek86] Kaczmarek, T., Ray Bates and Gabriel Robins. "Recent Developments in NIKL," *Proceedings, AAAI '86*. pp. 978–985, Philadelphia, PA., August, 1986.

[1] Mackinlay, Jock D. "Automatic Design of Graphical Presentations." Ph.D. Thesis, Department of Computer Science, Stanford University, Stanford, CA, December 1986.

[McAllester82] McAllester, D. A. "Reasoning Utility Package User's Manual," Massachusetts Institute of Technology, AI Memo 667, Cambridge, MA., April, 1982.

[Sondheimer86] Sondheimer, Norman K. and Bernhard Nebel "A Logical-Form and Knowledge-Base Design For Natural Language Generation," *Proceedings, AAAI '86*, pp. 612–618, Philadelphia, PA., August, 1986.

[Sutton78] Sutton, J. and R. Sprague, "A Study of Display Generation and Management in Interactive Business Applications," Technical Report RJ 2392 (31804), IBM Corporation, San Jose, CA., Nov., 1978.

[Vilain85] Vilain, M. "The Restricted Language Architecture of a Hybrid Representation System," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, CA., August, 1985.

[Yen88] Yen, J., R. Neches and R. MacGregor, "Classification-Based Programming: A Deep Integration of Frames and Rules," Research Report ISI/RR-88-213, USC/ISI, Marina del Rey, CA., March, 1988.